

```
1 void manipulate () {
2     // code
3     synchronized (syncObj) {
4         // critical section
5     }
6     // code
7 }
```

***FIG. 1***

1 *shared objects:*

2 flag[0..n-1] in {-1, .. n-2}

3 turn[0..n-2] in {0, .. n-1}

4 flag[i] <- -1

5 <entry>

6 for k=0 to n-2 do

7     flag[i] <- k

8     turn[k] <- i

9     while     (there exists j!=i,

10               flag[j] >= k and

11               turn[k] = i) do nothing

12

13 <critical section>

14

15 <exit>

16 flag[i] <- -1

**FIG. 2**

Process A	Line No.	Process B
<entry>	1	<entry>
for k=0 to 0 do	2	for k=0 to 0 do
flag[0] <- 0	3	flag[1] <- 0
turn[0] <- 0	4	turn[0] <- 1
while (there exists	5	while (there exists
j != 0, flag[j] >= 0	6	j != 1, flag[j] >= 0
and turn[0] = 0)	7	and turn[0] = 1)
do nothing	8	do nothing
<critical section>	9	<critical section>
<exit>	10	<exit>
flag[0] <- - -1	11	flag[1] <- - -1

**FIG. 3**

```
<entry>

for (int k=0; k <= (to numThreads-2); k++) {

    flag[tid] = k; // "tid" is "i" in Peterson's
    algorithm

    turn[k] = tid

    toYield = 0;

    do {

        if (toYield++ >= YieldCount)

            Thread.yield();

        allflag = false;

        for (int j=0; j < numThreads; j++) {

            if (j==tid)

                continue;

            allflag = allflag || (flag[j] >= k);

        }

        } while (allflag && turn[k]==tid);

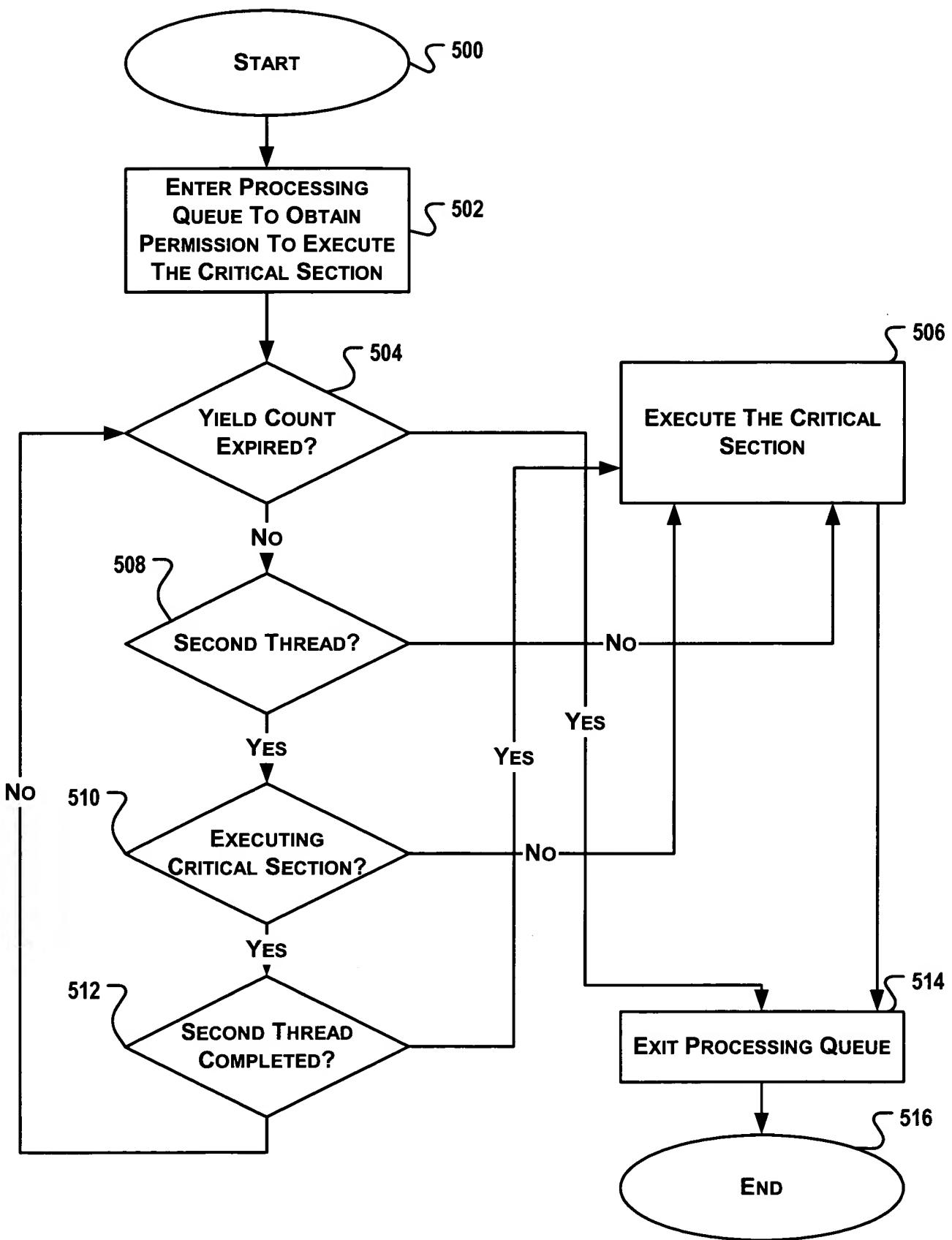
    }

//critical section

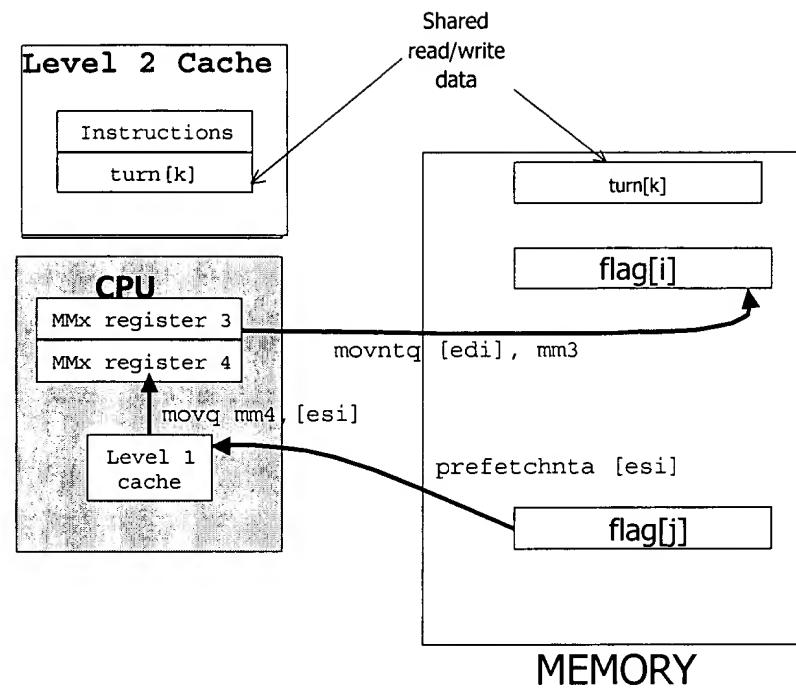
//exit

flag[tid] = -1
```

**FIG. 4**



**FIG. 5**



**Key:**

**prefetchnta** = Non Temporal  
**movntq** = Streaming Store  
**movq** = Normal Read or

**FIG. 6**